Homework #2

CS520 / KAIST Fall 2015

Due: October 14, 2015 2:30PM

1. Consider the following simple integer-expressions:

$$e \to \texttt{0} \mid \texttt{succ}\, e \mid \texttt{pred}\, e \mid \texttt{-}\, e$$

We define the expression's standard semantics as follows:

$$\texttt{0} \Rightarrow 0 \qquad \frac{e \Rightarrow n}{\texttt{-}\, e \Rightarrow -n}$$

$$\frac{e \Rightarrow n}{\texttt{succ}\, e \Rightarrow n+1} \quad \frac{e \Rightarrow n}{\texttt{pred}\, e \Rightarrow n-1}$$

where n is an integer and the unary and binary operations are the usual integer operations.

We also define a non-standard "strange" semantics as follows:

$$\texttt{0} \rightsquigarrow \odot$$

$$\frac{e \rightsquigarrow \odot}{\texttt{-}\, e \rightsquigarrow \odot} \qquad \frac{e \rightsquigarrow \oplus}{\texttt{-}\, e \rightsquigarrow \top} \qquad \frac{e \rightsquigarrow \top}{\texttt{-}\, e \rightsquigarrow \top}$$

$$\frac{e \rightsquigarrow \odot}{\texttt{succ}\, e \rightsquigarrow \oplus} \qquad \frac{e \rightsquigarrow \oplus}{\texttt{succ}\, e \rightsquigarrow \oplus} \qquad \frac{e \rightsquigarrow \top}{\texttt{succ}\, e \rightsquigarrow \top}$$

$$\frac{e \rightsquigarrow \odot}{\texttt{pred}\, e \rightsquigarrow \top} \qquad \frac{e \rightsquigarrow \oplus}{\texttt{pred}\, e \rightsquigarrow \top} \qquad \frac{e \rightsquigarrow \top}{\texttt{pred}\, e \rightsquigarrow \top}$$

where the special symbols represents subsets of integers as follows:

$$\begin{aligned} [\![\oplus]\!] &= \{n \in Z \mid n > 0\} \\ [\![\odot]\!] &= \{0\} \\ [\![\top]\!] &= Z \end{aligned}$$

Prove that: If $e \Rightarrow n$ and $e \rightsquigarrow \star$ then $n \in [\![\star]\!]$.

$\square$

2. Modified exercise 6.15 in the book "Glynn Winskel"

   Using the Hoare rules, prove that for integer *n*, *m*,

$$\{X = m \wedge Y = n \wedge Z = 1\}\ c\ \{Z = m^n\}$$

where c is the while-program

$$\texttt{while}\ \neg(Y = 0)\ \texttt{do}\ (Z := Z \times X;\ Y := Y - 1)$$

3. Consider the syntax of the language *E* for the “palm calculator”.

E	→	*n*	number
	|	*x*	variable
	|	*E* + *E*	addition
	|	*E* − *E*	subtraction
	|	let *x* = *E* in *E*	binding

We will implement this language *E* with < *S, E, C* >-machine. The < *S, E, C* >-machine is an abstract machine. *S* is a stack of values (ordered sequence). *E* is an environment (function: Variable → Value). *C* is a command sequence defined as following:

C	→	add. *C*	
	|	sub. *C*	
	|	bind(*x*). *C*	*; allocate local var x in E*
	|	unbind(*x*). *C*	*;deallocate most recently bound value x*
	|	push(*x*). *C*	
	|	push(*n*). *C*	
	|	ε	empty command

(a) Define the semantics of the language *E* and the command *C*.

(b) Define the compilation rules from *E* programs to *C* command sequences as we did in the class.

(c) State the correctness theorem of your compiler.

(d) Prove your theorem for the case **let x = E in E** .